2022

Temporal-difference comparison of learning methods for stock market prediction.

Maina, Stephen Gakuo
Strathmore Institute of Mathematical Sciences
Strathmore University

Temporal-difference Comparison of Learning Methods for Stock Market Prediction

Strathmore
UNIVERSITY

Stephen Gakuo Maina

Thesis presented in fulfillment of the academic requirement for the degree of Masters in Statistical Science of Strathmore University

September 2021

Declaration and recommendation

Declaration

I declare that this work has not been previously submitted and approved for the award of degree by this or any other university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Stephen Gakuo Maina

Signature .....

Date: 23-May-2021

Approval

The thesis of **Stephen Gakuo Maina** was reviewed and approved by the following:

Phd Elphas Luchemo Okango

Signature .....

Date: 23-May-2021

Strathmore University

Dr. Bernard Shibwabo

Director of Graduate Studies

Strathmore University

Abstract

Background: A Stock/Securities exchange is considered to be among the primary indicators' of a country's economic strength and development. Stock market prices are volatile in nature and are affected by factors like inflation, economic growth, etc. Prices depend heavily on demand and supply dynamics. Stock market price determination using ANNs has gained a lot of traction lately due to the obvious advantages this would represent to traders. Most methods in use today have largely been based on the feed forward algorithms, however, evolutionary techniques remain largely unexplored for this process despite their obvious robustness.

Method: Using data from the Nairobi Securities Exchange, and specifically the NSE20 Share Index, the project will seek to apply and compare traditional ANN techniques for stock market prediction against the relatively new evolution algorithms. The Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE) and a confusion matrix will be calculated for performance evaluation.

Results: The empirical results showed that the proposed evolutionary techniques out performed classic artificial neural networks methods-feed forward backpropagation.

Contents

List of Figures

List of Tables

Acknowledgment

The success and final outcome of this project required undivided attention and guidance from a whole host of people.

I would like first to thank my supervisor, Dr. Elphas Okango, who took interest in this research and made available his expertise that was invaluable in formulating this research project. Your insightful feed-back pushed me to sharpen my thinking and brought my work to a higher level.

I would like to acknowledge my course mates from the university for their wonderful collaboration. In addition, I would like to thank my wife for her understanding and patience as I took many hours huddled in a corner of our house working late hours on this project and my 3 Kids who always provided a timely call for me each time I wanted to put my feet up by telling me "daddy go and study"- I shall return the favour one day.

List of publications

Abbreviations

ANNs - Artificial Neural Networks

AR - Autoregressive Model

AP - Autoregressive Process

Backpropagation - Backward Propagation of Errors

BPN- Backward Propagation of Errors

DE - Differential Evolution

DL - Deep Learning

GA - Genetic Algorithm

GA-DE - Genetic Algorithm-Differential Evolution

I - Intergrated

IDE - Integrated Development Environment

LMA - Levenberg-Marquardt Algorithm

MA - Moving Average

MAE - Mean Absolute Error

ML - Machine Learning

MSE - Mean Squared Error

NN- Neural Network

PMM - predictive mean matching

ReLU - Rectified Linear Unit

RMSE - Root Mean Squared Error

SGD - Stochastic Gradient Descent

SVM - Support Vector Machines

Chapter 1

Introduction

1.1 Background to the study

According to (26) stock market prediction is the deliberate act of trying to determine the future value of a company's stock or other financial instrument traded on an exchange. Here an exchange or stock market refers to a regulated platform for trading a company's derivatives e.g shares at an agreed price (24).

A lot of effort has been made in the field of stock market prediction, and a raft of ideas and methods have been proposed and can be categorized into three broad schools of thought. The first category, believes that no investor can achieve beyond average trading advantages based on historical and present information. The second category,is that of rudimentary analysis, whereby industry experts undertake an in-depth study of macro-economic factors and prevailing financial conditions, this allows them to figure out the extent of correlation that may exist with changing stock prices. A third view is that of recurring patterns in the markets, these patterns can be identified and used for predictions (16).

However, these three techniques yield unreliable results mainly caused by their over reliance to human judgement, which fails to address the inherent nature of stock market indices that are by nature dynamic, non-linear, and non-parametric. (16).

To address these drawbacks, the recent trend has been to develop statistical models to forecast financial data, with various technologies such as ANNs, SVMs taking center stage. Forecasting is a major application area of ANNs algorithms. There has been an increasing interest in forecasting using artificial intelligent systems in recent years. The importance of forecasting as a subject is underscored by the diversity of its applications in a wide range of disciplines ranging from business to engineering. Having the ability to accurately predict the future (within a margin of error) is fundamental to many decision processes in planning, scheduling, purchasing, strategy formulation, policy making, and

stock price (2). Forecasting has been dominated by linear methods for many decades. Linear methods are easy to develop and implement and they are also relatively simple to understand and interpret. However, linear models have serious limitation in that they are not able to capture any nonlinear relationships in data (2).

The approximation of linear models to complicated nonlinear relationships is not always satisfactory. In the early days of forecasting, large-scale competitions (known as M-competitions) were organized to test the commonly used linear methods with real time series data. The mixed results obtained from these competitions, showed that no single linear model is universally the best, which we can now inter-prate as the failure of linear models in accounting for a varying degree of non-linearity that is common in real world data (17).

Stock market prediction is one field that is gaining a lot of traction lately from researchers due to its obvious commercial applications, the high stakes and the kinds of attractive benefits that it has to offer. (16).

1.1.1 Artificial Neural Networks Models

Artificial neural networks are self learning mechanisms that are able to work in tandem with input variables, a feature that allows them to handle large sets of data (1). They mimic the neural structure of the brain, by basically learning from experience, brains store information as patterns.

The principal advantage of artificial neural networks is their ability to find patterns and approximate any nonlinear function to a very high degree of accuracy(16), this in turn makes them ideal models compared to fore asters, this is because they poses an inherently nonlinear structure useful for capturing the complex underlying relationship in many real world problems (1). They are not only capable of finding nonlinear structures in a task, but can also be used to model linear processes. A number of researchers have shown the capability of neural networks in modelling linear time series (11).

1.2 Statement of the Problem

Backward Propagation of Errors algorithm requires that in all instances the gradient to be defined, in other words the task should be differentiable, but not all data in the context of the real world are, most practical problems have objective functions that are non-differentiable, noisy, are non-linear in nature and have many local minima. Faced with such problems it is impossible to solve them analytically. Differential Evolution algorithm can be applied to find approximate solutions to such problems.

1.3 Objectives of the Study

1.3.1 Main Objective

The main objective of this project is to review and compare existing Artificial Neural Network Algorithms for stock market prediction and gauge their efficacy.

1.3.2 Specific Objective

1. Evaluate the feasibility of Machine Learning Algorithms to predict stock market prices in order to make more informed and accurate investment decisions.

2. Assess the accuracy and performance of ANNs learning algorithms in stock market price prediction.

Chapter 2

Literature review

2.1 Introduction

A significant challenge facing prospectors is to accurately predict the index of a stock market in a limited amount of time in order to make credible investment decisions that will yield profits on invested monies. Historical data points to the fact that history repeats itself and thus an investor can use this proxy as a prediction equivalent, but this association is feeble and rarely yields tangible results.

To this effect various techniques have been formulated to investigate and come up with more scientific models to explain the relationship between historical and present data to determine tomorrows prices. These techniques can classified into two broad arenas: Fundamental Analysis- mainly focuses on external economic factors and prevailing events, it's basis being the law of supply and demand which in turn makes prices go up, low or stay the same. Technical Analysis- has it's focus set on market movements, through the use of data analysis to find patterns that can be used to forecast future trends on prices. the two main techniques are different in that the former studies the cause of market action, while the latter looks at effects(12). They are however disadvantaged by the nonlinear nature of most data (1).

More recently, artificial intelligence techniques and hybrid intelligent systems have been mooted to tackle limitations of traditional statistical methods in nonlinear and time variant problems of finance data. One of the artificial intelligence technologies is artificial neural networks (ANN), which is a supervised, self-learning technique, that can find the smooth approximation between input and output data and recognize new patterns not present in current data(12).

BPN neural network is the widely used method for stock market price forecasting, it applies a multilayer model whose main principle lies in spreading the internal network error by applying the gradient descent technique to lower the empirical network error.

The performance of this method is affected by the choice of hidden layers-how many neurons will be selected, the transfer function between layers and the choice of parameters (12).

2.1.1 Traditional Forecasting Methods-ARIMA

The ARIMA model was introduced by Box and Jenkins(1976) and has shown efficiency in generating short- term forecasts. In this model, the future value of a given variable is a linear combination of past values and past errors (3). Its an univariate method as it uses previous values of a variable to predict future values of the same variable.

The model is comprised of three component functions: AR (p), the number of lag observations or autoregressive terms in the model; I (d), the difference in the nonseasonal observations; and MA (q), the size of the moving average window. The model order is depicted as (p,d,q) with values for the order or number of times the function occurs in running the model, a zero value is admissible.

The model uses differenced data to make the data stationary, that results in consistency of the data over time. This function mitigates the effect of trends or seasonality, such as market or economic data.

However, the model suffers from major draw backs in prediction in that it makes an assumption of constant variance, most financial data are however volatile and this feature of the data can't be fulfilled under this assumption, in addition to this, the symmetric joint distribution requirement of the stationary models does not fit data with strong asymmetry a feature all to common in financial data.Thus the model is more suitable for use with data that exhibit a negligible amount of sudden burst of large amplitude at irregular time epochs. These limitations of the integrated ARMA models in forecasting time series data lead us to models where we can retain and allow data to be non-Gaussian or abandon the linearity assumption.

2.1.2 Modern Forecasting-ANN

The first step toward artificial neural networks came in 1943 when (18), a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. They modeled a simple neural network with electrical circuits.
Donald Hebb, in his 1949 book *Organization of Behavior*, pointed out that neural pathways are strengthened out each they are used.
Artificial Neural Networks use and application has gained a lot of interest in the recent past, their applicability in the arena of prediction and classification has endeared them even further. These networks are practical attempts aimed at mimicking the human brain by design and implementation, normally, the brain is capable of recognizing patterns even in the presence of statistical noise (unexplained variability in a data sample), form abstract ideas, have fault tolerance and be able to recall memories that allow it to make judgments on situations never before encountered (27). These networks are now being co-opted in fields where traditional statistical methods have been widely used before, for instance (8) used them in a classification problem to identify underwater sonar contacts, while (5) used them to predict heart problems in subjects by visual interpretation of nuclear images. Neural networks have also been used in speech recognition (14), in application of time series, neural networks have been used in the prediction of stock market performance as outlined by (10).
These problems are usually solved by the application of basic statistical models such as logistic regression, discriminant analysis and whole host of other classifiers such as the Bayes. A handful of researchers have carried out studies to compare these statistical methods and neural networks with the main aims being to gauge performance on application to specific problems.

2.1.3 Related Works

The main focus of predictive systems currently is to develop machine learning methods with higher accuracy and efficiency for financial markets.
(6) used an ensemble system anchored on genetic algorithm. The system consisted of 10 SVM classifiers, each classifier had it's own set of inputs(technical indicators) and

parameters, their outputs were combined by majority voting. The genetic algorithm was used to select best features(variables) and parameter optimization for each SVM classifier in stock market price prediction for the Sao Paulo stock exchange. Their experimental results showed that their proposed model was more accurate as opposed to methods such as bagging, SVM, Random Forest with the only drawback being the length of time it took the model to run.

(21) proposed the use of hybrid forecasting models to predict (t+1) day closing price, they used technical indicators as inputs and applied support vector regression in conversion of technical indicator data, to predict the closing day price of the (t+1) day they applied the following models support vector regression, ANN and random forest and found out that the accuracy of two stage models was higher than that of single models when the number of predicted points increased.

(28) proposed a forecasting model by applying the genetic algorithm in sequential steps, first to get optimal weights for backpropagation network, then they used back propagation neural network, after that they trained and tested the network using optimal weights of the best fit chromosome, to predict close price of Shanghai index along ten days. GA was then used in order to improve the speed of convergence and to overcome overfitting problem because, as known BPN has two drawbacks, it falls easily into local minimum and has slow convergence. Also they used BP single model to compare it with the GA-BP proposed model results. These team proved that experimental results of the GA-BP had achieved the satisfactory accuracy of Shanghai composite index prediction.

(Jena and Padhy) proposed a hybrid forecasting model which entailed the application of a genetic algorithm to choose the best parameters for Support Vector Machine classifier, and proved that the proposed hybrid model GA-SVM is better than the single model of support vector machine (SVM) in predicting the prices of the Indian stock market indexes. The GA-SVM model improved the prediction accuracy as measured through DS, MAE and MSE.

We intend to add to this literature by way of contrasting a simple statistical regression model against a neural network tool, linear models have been used extensively for forecasting due to their ease in developing, implementing and at best require basic knowledge of statistics to interpret and understand.

2.1.4 Optimization Algorithms

Most deep learning algorithms and methods involve some sort of optimization, optimization refers the process of minimizing or maximising some function $f(x)$ by altering the constituent variable x (7). The function we want to minimize or maximize is referred to as the objective function, when we are specifically minimizing it we refer to it as the **cost, loss, or error** function.

To train neural networks for these tasks we apply optimization algorithms which are basically learning procedures which look for a set of parameters for which the loss index/function for the network will be at a minimum. There exists a large number of optimization algorithms each with it's own characteristics and numerical precision;

- Learning problem algorithm: measures the performance of a network on a data set and is formulated in terms of a minimization of a loss function, $\boldsymbol{f}$, this loss function is composed of a regularization term that is used to control over fitting by controlling the complexity of the network and an error term that evaluates how well the network fits the data set.

- Gradient descent algorithm: it is a first order method that requires information from the gradient vector. For this method we start at a point $x^{(0)}$ until a stopping criteria is meet. That is, we move from $x^{(i)}$ to $x^{(i+1)}$ in the training direction $w^{(i)}$=-$y^{(i)}$. All we are doing in this method is iterating $x^{(i+1)} = x^{(i)} - g^{(i)}n^{(i)}$ for i=0,1,..., the parameter **n** is the learning rate whose value can either be set to a fixed value or calculated by one-dimensional optimization at each training step. This method is best applied when we have a huge neural network with thousands of parameters, since it stores the gradient vector as opposed to the hessian matrix.

- Newton method: considered to be a second order algorithm as it makes use of the Hessian matrix, it's purpose is to use the second order derivatives of the loss function to find more efficient training direction, mathematically we have this algorithm expressed as follows,if we let;

$$f(W^{(i)}) = f^{(i)}, \forall f(W^{(i)}) = g^{(i)}$$

and

$$Hf(W^{(i)}) = H^{(i)}$$

$H^{(0)}$ is the hessian matrix of $\boldsymbol{f}$ evaluated at a point $W^{(0)}$, if we set g to be equal to zero as the minimum, we get the equation

$$g = g^{(0)} + H^{(0)}.(W - W^{(0)}) = 0$$

with an initial vector $W^{(0)}$, this method iterates

$$W^{(i+1)} = W^{(i)} - (H^{(i)-1}.g^{(i)})n$$

, the value n ensures that the newton step moves towards a minimum if the hessian matrix is not positive definite.

- Conjugate gradient: is an intermediate method between gradient descent and newton's method that was formulated to deal with slow convergence issue that ails the gradient descent optimization algorithm, and also avoids some of the information needs associated in the evaluation, storage and inversion of the Hessian matrix a prerequisite of the Newton method.
  In this training algorithm, the search occurs along conjugated directions with respect to the Hessian matrix, assume starting with an initial parameter vector $W^{(0)}$ and an initial training direction vector $t^{(0)} = -g^{(0)}$.
  The method proceeds to construct successive training directions, expressed as follows

  $$t^{(i+1)} = g^{(i+1)} + d^{(i)}.\gamma^{(i)},$$

  for i=0,1,... and γ is referred to as the conjugate parameter. **NB**:the training direction is reset to negative of the gradient at regular intervals for all conjugate gradient algorithms and the parameters improved by applying

  $$W^{(i+1)} = W^{(i+1)} + d^{(i)}.\eta(i),$$

  η is the training rate calculated by line minimization.

- Quasi-Newton Method: was formulated to deal with the high computational re-

quirements of Newton method in evaluating the Hessian matrix and its inverse. This method comes up with an approximation of the inverse Hessian at each iteration of the algorithm, by using information derived by the first derivative of the loss function. Thus, the idea behind this method is that of approximating the Hessian by use of another matrix by using only the first partial derivatives of the loss index. It is expressed as;

$$W^{(i+1)} = W^{(i)} - (G^{(i)}.g^{(i)}.\eta^{(i)})$$

The training rate η can either be set to a fixed value or found by line minimization.

- Levenberg-Marquardt algorithm: was made to work with loss functions that take the form of a sum of squared errors. it utilizes the Jacobian matrix and the gradient vector. We can express the sum of squared errors of a loss fuction as follows

$$f = \sum_{i=1}^{n} e_i^2$$

n is the instances in data.

The Jacobian matrix of the loss index contains the derivatives of the errors concerning the parameters.

$$J_{i,j} = \frac{\delta e_i}{\delta W_j}$$

for i=1,...,k and j=1,...,l, with k being the number of instances in the data set and l the number of parameters in the neural network. The jacobian matrix is of dimension **k.l**. To compute the gradient vector for this method we:

$$\nabla f = 2J^T.\epsilon$$

with ϵ being a vector of the error terms. The Hessian matrix under this method is derived by the following expression

$$Hf \approx 2J^T.J + \lambda I$$

, here lambda is known as the dampening factor ensuring the Hessian is positive and **I** is the identity matrix.

Performance comparison of the above algorithms is depicted in the chart below, the slowest method is the gradient descent method requiring less memory but a slow convergence speed, LMA is fast but requires a substantial amount of computational memory.

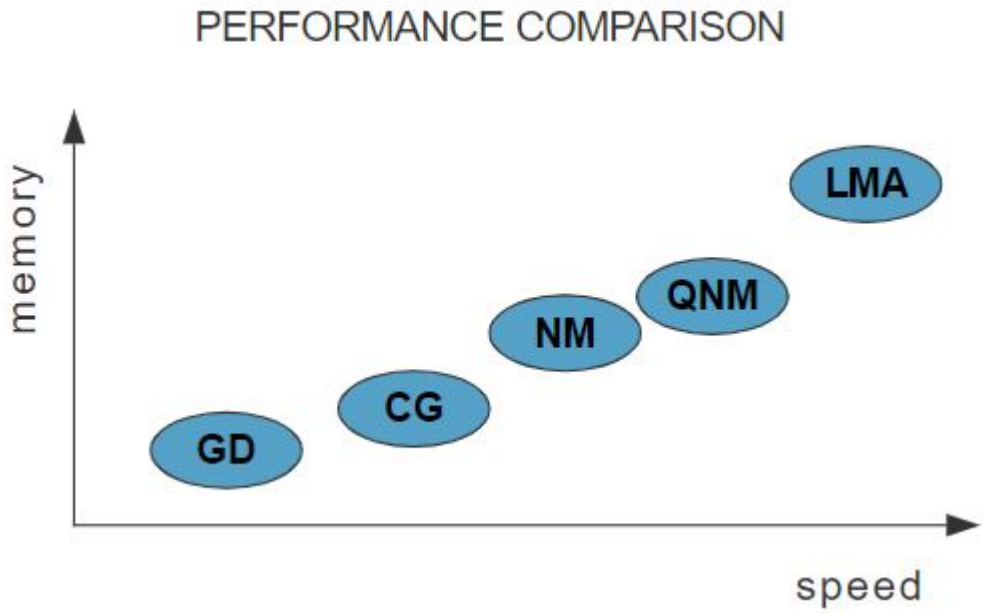


Figure 2.1: Optimization algorithms compared

In conclusion the choice of optimization algorithm in our neural network is dependent on the number of parameters, if in thousands gradient descent or the conjugate gradient algorithms are preferred, for a few hundreds of parameters the best choice will be the LMA.

For this research the gradient descent optimization will be applied to the NSE20 share index during BPN and GA-DE optimization due to the fact that it utilizes a gradient vector.

Stock prices can be regarded as a random time sequence characterized by unexplained variability/noise, artificial neural networks are large-scale nonlinear parallel processing systems that depend on their own intrinsic link data, they provide methods and techniques that can be used to approximate any nonlinear continuous function, without making any assumptions about the nature of the generating process (22).

Chapter 3

Methodology

3.1 Introduction

Machine learning algorithms can be viewed as statistical computer programs that automatically improve in performance as they acquire more and more experience, they work by building mathematical models based on a given set of data, and use this models synthesized from data to make predictions or decisions about scenarios not experienced before hand. For instance an algorithm intending to perform classification, might be taught to identify the features of an object and assigns a weight to that attribute based on how valuable it is in correctly identifying the object, this allows it, to within a certain degree of accuracy identify a similar object in the future.
We have two main problems that we aim to resolve under the category of supervised machine Learning;

i. Classification: algorithms in use this field aim to estimate the mapping function $f(x)$ from some input variable(s) x to a discrete or categorical output variable(s) y.

ii. Regression: while in regression we employ algorithms to learn and estimate the mapping function $f(x)$ from the input variable(s) x to a numerical or continuous output variable(s) y.

In this research our focus will be on regression using artificial neural networks. A simple regression equation is of the form:

$$\hat{Y} = w_1X_1 + w_2X_2 + ... + w_nX_n + b$$

where: $X_n's$ are a set of inputs, $W_n'S$ correspond to weights, b is the bias, $\hat{Y}$ is the the output.

3.2 Artificial Neural Networks Architecture

The basic computational unit of the human brain is the neuron, these neurons number 10^{11} in the human brain, they work parallel to one another but are highly interconnected for information sharing. For ANNs the single neuron network or the perceptron is the fundamental building block. In essence, we have in a neural network a network of interconnected processing units, such networks allow us to extract meaningful information as well as detect hidden patterns from complex data sets.

The single neuron network is the basic unit of an artificial neural network, it is referred to as the **perceptron**.

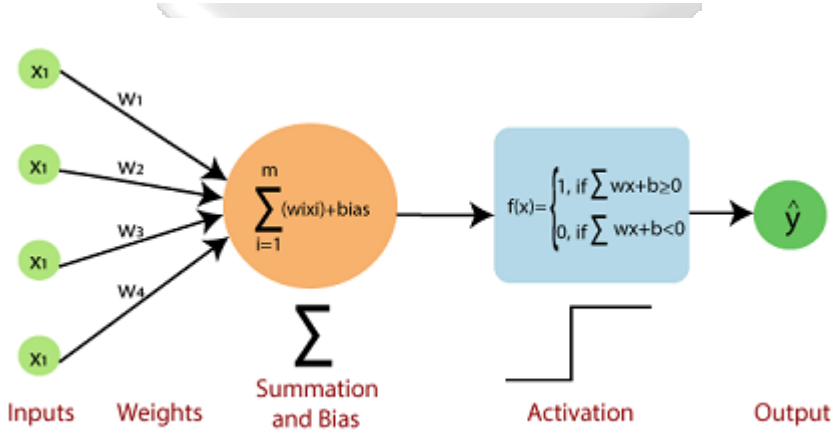


Figure 3.1: A Perceptron: Wiki-Images

- Where the x_i's are a set of inputs that are multiplied by a corresponding weight w_i and the result summed up.
- a bias term w_0 is then added to the weighted sum and passed through a non linear activation function to yield the desired output $\hat{Y}$.

Therefore, to compute the output of a perceptron: we first compute the dot product, we then add a bias term then take a non-linearity (Activation Functions in Deep Learning).

3.2.1 Activation Functions Choice

The purpose of activation functions is to introduce non-linearity's that allow us to approximate arbitrarily complex functions in a network, this is because in real world

applications of ANNs, data is almost always non-linear.

Activation functions are in essence mathematical equations that determine the output of a neural network, this functions are attached to neurons and determine if each neuron will be fired or activated, their main attribute is that they must be computationally efficient as they are calculated across a large of neurons in a given data sample. Their are awash of activation functions to choose from in the application of Neural Networks. This choice in the hidden layer controls how well the the network learns from the training data set and consequently, the choice in the output layer is determined by the type of predictions the model will make.

A differentiable nonlinear activation function is typically applied in the hidden layers of a neural network as it allows the model to learn more complex functions (4).

3.2.1.1 Sigmoid Function

A mathematical function that produces a sigmoidal curve, makes a model logistic in nature for predicting probabilities since it squashes the input to values between 0 and 1.

$$f(x) = \frac{1}{1 + \exp(-x)}$$

However, this method suffers from the following disadvantages;

1. since it uses a logistic model, the computations are time consuming and complex,
2. it causes gradients to vanish and no signals pass through the neurons after some point of time
3. It is slow in convergence
4. It is not zero centered Ciaburro and Venkateswaran

3.2.1.2 Hyperbolic Tangent

Is a scaled sigmoid function with a stronger gradient, nonlinear in nature, defined in the range of values (-1,1). It is calculated as follows;

$$f(x) = tanh(x)$$

(4)

3.2.1.3 Rectified Linear Unit-ReLu

The most widely used activation function as it is relatively easy to compute, has non-linearity at k=0. It is defined as follows;

$$f(k) = 0, k < 0, \text{and k when k is greater or equal to 0}$$

. The range of output is between zero and infinity.
The choice of activation has to be robust enough to satisfy the following, it should be differentiable, that is, it should cause gradients to disappear, it should be fast processing and simple, and should not be zero centered.
In this research we will use the ReLu activation funtion, **justification:** it is simple and fast to process, it does not suffer from the vanishing gradient problem exhibited by *sigmoid* and *tanh* functions. it is widely used for DL algorithms(4).

3.2.1.4 linear function

Is the simplest activation function, and is commonly used for the output layer activation in most deep learning neural network problems, is characterised by the output being the same as the input. Derived as follows $y = f(x) = x$ and is defined in the range of $-\infty$ to ∞ (4). For this research the linear activation function will be applied at the output layer for the BPN algorithm.

3.2.2 Overfitting Mitigation

A major consideration when fitting an ANN is overfitting mitigation, basically, overfitting is a modelling error that occurs when a model is too closely aligned to a limited set of data points, which causes such a model/network to perform poorly on an unseen data set. Several mitigation techniques for this error exist as detailed below;

- Model simplification- main cause of overfitting is a complex model with too many parameters, this means a model complexity can be reduced by reducing the number

of neurons or removing layers that in turn makes the model small, this ensures variables that have maximum variability on the outcome are feed to the network.

- Early stopping- simply stop as soon as the validation loss rises above a maximum threshold, these methods work to improve a model fitness with each iteration by providing rules that guide on how many iterations can be done prior to the model overfitting.

Figure 3.2: Early Stopping-Wiki Image

- Regularization- a techniques that is also used to reduce a model complexity by adding a penalty term to the loss function. Most common regularization methods are L1 and L2.

  – L1 Aims to reduce the absolute value of weights, it's mathematically represented as;

  $$L(x, y) = \sum_{i=1}^{n}(y_i - h_0(x_i))^2 + \lambda \sum_{i=1}^{n} |\theta_i|$$

  We finally end up with a model that is simple, interpretable and robust to outliers.

  – L2 In turn aims at minimizing the squared magnitude of the weights, it's mathematically represented as;

  $$L(x, y) = \sum_{i=1}^{n}(y_i - h_0(x_i))^2 + \lambda \sum_{i=1}^{n} \theta_i^2$$

  Allows us to learn complex data patterns.

- Dropouts Technique- is a regularization technique that modifies the neural network by randomly dropping neurons during model training after each iteration. The overarching idea here being that when a set of neurons are dropped, the subsequent run is like training a new network all together, thus the net effect of dropouts will reduce overfitting by ensuring the network is not over reliant on any one path, and therefore develops a whole ensemble of different paths.

- Batching- Computing the gradient is computationally expensive, more so with modern data sets that are ever lengthy, thus it is not feasible to use all data points to compute the gradient, to overcome this we batch our data into mini batches of say **K** data points, from this we can get a true estimate of the gradient by averaging the gradients from each of these sets of data points (4).

For this research the early stopping overfitting method was applied for both the BPN algorithm and Genetic Algorithm method.

3.2.3 Loss Function

The loss function aspect is applied during model training, we wish to find the neural network weights **W**, that by which we achieve the lowest empirical loss over the training data set. Loss optimization formula will be as below;

$$W^* = argmin \frac{1}{n} \sum_{i=1}^{n} L(f(x^i; W), y^i)$$

In summary, the loss function of the gradient descent method tells us how good the model is at making predictions for a given set of parameters while, it's slope tells us how to update our parameters to make the model more accurate (23).

3.2.4 Model hyperparameters to consider

As alluded to earlier hyperparameters in ANNs are crucial variables that the statistician has to decide on before execution of the network, such as;

Batch - which defines the number of samples to work through before updating the

internal model parameters. At the end of each batch, a comparison is made between the predictions and the expected output variables, an error metric is then calculated, from this a decision is made to update the algorithm and consequently the model.

Epoch- defines the number of times a learning algorithm will go through the entire training data set, it gives each sample in the training set an almost equal chance to update the internal model parameters. An epoch is made up of one or more batches but it should be sufficiently large to allow a learning algorithm run until the model error is minimized.

3.2.5 Choice of layers of neural network

By design most neural networks have three types of layers;

1. Input layer - every neural network has exactly one input layer, the number of neurons is completely and uniquely determined by the structure of the training data. Specifically, the number of neurons/nodes comprising this layer is equal to the number of features in your data, in some configurations though we can add one additional node for a bias term.

2. Hidden Layer - is the most dynamic layer in a neural network, it is referred to as hidden since it is not observable and has to be learned, if the data under investigation is linearly separable then a hidden layer is not needed, the optimal size of the hidden layer is usually between the size of the input and size of the output layers, (i) number of hidden layers equals one; and (ii) the number of neurons in that layer is the mean of the neurons in the input and output layers (20).

   For this research the number of neurons was calculated as follows;

$$Num_{neurons} = \frac{2(N+1)}{3}$$

   , N is the number of neurons in the input layer, applying this equation gave me **3** neurons in the hidden layer for the BPN algorithm in this research.

3. Output layer - similar to the input layer every neural network has exactly one

output layer, its size is determined by the model configuration, that is, if the NN is a regressor, then a single node will suffice for this layer. In case the NN is a classifier, it will also use a single node, if softmax is used then the output layer will have one node per class label in your model.

The choice of layers in a ANN determines it's accuracy and convergence speed.

3.3 Algorithms under consideration

The algorithms under review in this research try to improve this $\hat{Y}$ by adjusting the weights iteratively and calculating a loss function, given by
$MSE = \frac{1}{2} * [(Y_{actual} - Y_{pred})]^2$.
I propose to use the Backpropagation and the Genetic algorithm-Differential Evolution method to minimize this loss function.

3.3.1 Backward Propagation of Errors

Backpropagation is a supervised learning algorithm for ANNs that employ the gradient descent method to reduce some error function called the loss function, done by repeatedly adjusting the weights of the connections in a network. (23).

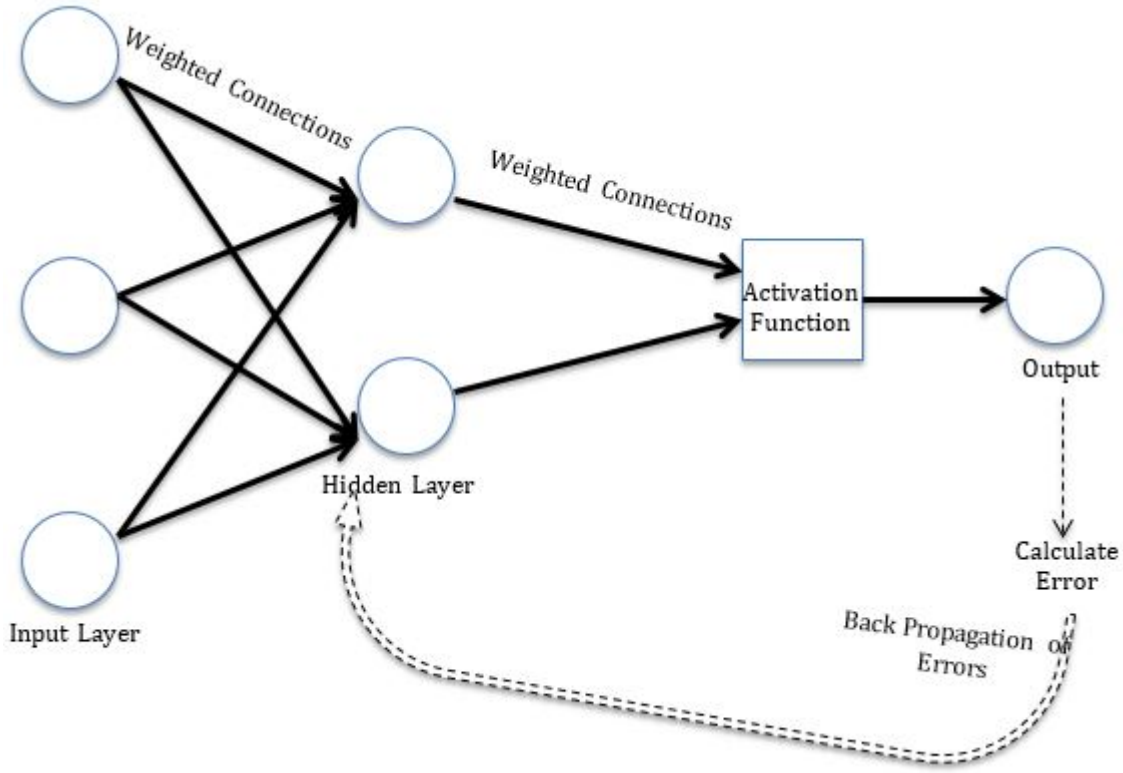


Figure 3.3: A simple BPN Architecture

3.3.1.1 Gradient Descent

Gradient descent, is an optimization method that is used to update model paramaters of a feed-forward back-propagation algorithm, it minimizes the loss function by iteratively moving in the direction of the steepest descent as defined by the negative of the gradient. Our main aim being to achieve a predicted value that is close as possible to the original value. This method assumes a global minimum value that must be found in order to find the minimum loss function (23). This is only achievable if our loss function graph assumes a parabolic shape. It is important to note that predicted values are dependent on the weights, at the initial stage, the model assigns arbitrary weights to features until a minimum loss function is achieved.

So, how does the the algorithm know; it needs to optimize weights to minimize the error, in essence what we need to check is how the error varies with weight.

To do this we find a derivative of error with respect to weight, this is what we refer to as a gradient;

$$Gradient = \frac{\delta E}{\delta W}$$

The negative of the gradient will always show the direction along which the weights should be moved in order to optimize the loss function, once the movement direction is established, we then have the model determine the rate of movement. This rate is referred to as the **learning rate**, denoted by α. This in essence is the distance to be moved form point a to a new location b.

$$dx = \alpha * |\frac{\delta E}{\delta W}|$$

A high learning rate α implies that we cover more ground per step but we risk overshooting our lowest point, whereas a low learning rate is more precise but computationally expensive. Our new weight is thus given by;

$$W_j(new) = W_j(old) + \alpha * e(i) * \frac{\delta Y}{\delta W_j}$$

(19)

3.3.1.2 Choice of learning rate, η

The learning rate eta (η) is the most important hyper-parameter (a hyper-parameter is a parameter that needs to be chosen by the programmer before executing a ML algorithm), several methods exist that aid in the determination of this rate;

Learning rate annealing- presupposes starting with a high learning rate and then gradually reducing the learning rate linearly during training, with the rate decreasing to a value close to 0. The intuition behind this approach is that we'd like to move quickly from the initial parameters to a range of optimal values but then we'd like a learning rate small enough that we can explore the "deeper, but narrower parts of the loss function". Step decay is the most popular form of learning rate annealing, here the learning rate is reduced by some margin, usually a percentage after a fixed number of training epochs.

Cyclical learning rates- (25) in her paper proposed that instead of monotonically decreasing the learning rate it should be varied cyclically between two reasonable boundary

values. The general schedule can be written as

$$\eta_t = \eta_{min} + (\eta_{max} - \eta_{min})(max(0, 1 - x))$$

where x is expressed as

$$x = \left| \frac{iterations}{stepsize} - 2(cycle) + 1 \right|$$

the cycle is calculated as follows

$$cycle = floor\left(1 + \frac{iterations}{2(stepsize)}\right),$$

where η_{min} and η_{max} define the bounds of our learning rate, iterations represents the number of completed mini-batches, step size defines one half of a cycle length, $1x$ should always be positive.

Stochastic Gradient Descent with Warm Restarts (SGDR)- is where an annealing schedule is combined with periodic "restarts" to the initial starting learning rate η the restart is warm as the optimization does not start from scratch but from the parameters to which the model converged during the last step (15). This method relies on an aggressive cosine annealing function to lower the learning rate, the function varies between -1 and 1, mathematically it can be expressed as;

$$\eta_k = \eta^i_{min} + \frac{1}{2}(\eta^i_{max} - \eta^i_{min})\left(1 + cos\left(\frac{T_{current}\pi}{T_i}\right)\right)$$

, where η_k is the learning rate at time-step k , η^i_{min} and η^i_{max} define the range of desired learning rates, $T_{current}$ represents the number of epochs since the last restart, calculated at every iteration and can only take on fractional values, and T_i is number of epochs in an cycle. Empirically it has been found that SGD with warm restarts requires fewer epochs up to 50% fewer than learning rate annealing method and achieves comparable or better performance.

To select a good learning rate might mean starting off with different values and seeing which one gives you the best loss without sacrificing the speed of training. We shall implement the BPN algorithm in R by the use of the package **neuralnet**.

3.3.2 Genetic Algorithm - Differential Evolution

Genetic algorithms are based on the biological theory of evolution by natural selection formulated by Charles Darwin, he postulated that organisms evolve over generations, and that they do this by inheriting physical or behavioral traits from a previous generation through the process of natural selection. This is the process through which living organisms change and adapt. Therefore, individuals with better suited traits are more likely to survive and reproduce, and through this favorable traits are thus transmitted from one generation to the next.

GA in ANN are an implementation of this evolutionary technique observed in nature, they start with a set of candidate solutions referred to as a population, solutions in a population are then used to form a new population, the idea being that this new population will be superior to it's predecessor, the selection is based on a fitness criteria, the process is repeated until some user defined condition is meet.

The basic parameters of a genetic algorithm are as follows;

- Initialization: a random population of chromosomes of size n is generated, these are the candidate solutions to a problem.
- Fitness: each chromosome in the populous, is evaluated for fitness
- Birth of a new population: a new population is created by following the steps below
  - selection: an encoding is chosen
  - crossover: uses genes from parent chromosome to create a new offsprig
  - Mutation: prevent solutions from falling into a local minima prematurely
  - Acceptance: places an offspring into a new population
- Replace: Use new generated population for a further run of algorithm
- Testing: stop if criteria is meet, and return the best solution in current population, else loop to fitness step.

Problem solving in this aspect can be of regarded as looking for extreme of a function. Some function will be given and GA will try to find a minimum of the function.

In this research I am proposing to use Differential Evolution, which is a type of genetic algorithm, that use biologically inspired techniques to solve optimization problems that seek to find the best solution from all feasible solutions over a continuous search space, it does this by improving a problem by iteratively trying to improve a candidate solution based on an evolutionary process of crossover, mutation and selection over the course of successive generations (9). This algorithm makes few or no assumptions about the underlying optimization problem. It can find the true global minimum despite the initial parameter values, it does not use the gradient of the problem being optimized which means it does not require the problem at hand to be differentiable.

3.3.2.1 How does it work?

A. Initialization: it sets a population vector of candidate solutions of size **NP**; candidate solutions or agents refer to a set of all possible solutions available for an optimization problem that satisfy the problem constraints.

$$X_{i,G} = [X^1_i{}_{,G} .....X^D_i{}_{,G}]$$

where i=1,...,NP represents the total population. The parameter values are selected randomly and uniformly between X_{min} and X_{max}. To generate the initial value of the j^{th} parameter for an individual i at a generation Y=0, we sample;

$$X^j_{i\,0} = X^j_{min} + rand(0,1) * (X^j_{max} - X^j_{min})$$

B. Mutation: By applying the mutation operator these agents are then moved around in a search space by mathematical formulae to combine the positions of existing agents from the population, one mutant vector $V_{i,G}$ is produced, that is, for each target vector $X_{i,G}$ for generation **G**,it's mutant vector is

$$V_{i,G} = [V^1_i{}_{,G} ...V^D_i, G]$$

. Mutations are small random tweaks in the chromosome aimed at getting new solutions, types of mutation include;

- Bit flip mutation- one or more random bits are selected then flipped, mainly applied in binary encoded genetic algorithms.

- Random resetting- mainly applied in integer representations, a random value from the set of permissible values is assigned to a randomly chosen gene.

- Swap Mutation- 2 positions on the chromosome are chosen at random, the values at this positions are then interchanged, commonly applied in permutations based encoding.

- scramble mutation- from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.

- Inverse Mutation- a subset of genes are chosen, then the entire string in that subset is inverted.

C. Selection: We then compare the target vector $X_{i,G}$ with the trial vector $U_{i,G+1}$, the one with the lowest function value is assimilated into the next generation, that is, we accept an agent if the new position is an improvement, it then forms part of the population, else we discard this agent(s). DE does these repeatedly until we get a satisfactory solution which is however not guaranteed. Lets assume that $f : R^n \rightarrow R$ is a function that we plan to reduce/minimize. This function, takes a candidate solution as an argument in the form of a vector of $\mathbb{R}$, it the outputs a $\mathbb{R}$ which is now a measure of fitness of the given candidate solution. The gradient of this solution is not known. Thus, the aim is to find a solution $\mathbf{m}$ for which $f(m) \geq f(p)\ \forall\ \mathbf{P}$ in the search-space, this implies that $\mathbf{m}$ is the global minimum.
Their several ways to implement selection in genetic algorithms;

- Tournament selection- several tournaments are played among a few individuals whose selection is done randomly prior, the winner of each tournament is selected to be in the generation.

- Roulette wheel and proportionate selection- every individual can become a parent with a probability which is proportional to its fitness. Thus, fitter individuals have a higher chance of mating and propagating their features to the next generation,

the selection strategy applies a selection pressure to the more fit individuals in the population, evolving better individuals over time. It's implemented as follows;

  - Calculate S = sum of a finesses.
  - Generate a random number between 0 and S.
  - Starting from the top of the population, keep adding the finesses to the partial sum P, till $P < S$.
  - The individual for which P exceeds S is the chosen individual.

- Rank Selection- works with negative fitness values and is mainly used when the individuals in the population have very close fitness values, whereby, each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent, consequently, it leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selection. To overcome this, the concept of fitness is abandoned when selecting a parent and rank every individual in the populous according to their fitness, the higher ranked individuals are then preferred.

- Steady state selection- we use a few good chromosomes to create new off springs after every iteration, the bad chromosomes are discarded and replaced with the new offspring, the rest of the population migrates over to the next generation without the need of going through the selection process.

D. Recombination/Crossover: A trial vector $U_{i,G}$ will be generated from the elements of the target vector pair and its corresponding mutant vector this after going through the recombination process.

$$U_{i,G} = [u^1_i{}_{,G}\,, u^2_i{}_{,G}\,, ..., u^D_i{}_{,G}\,]$$

In GA crossover is applied with high probability and is used to create new solutions from existing ones after the application of the selection operator. Crossover mechanisms include

- One point crossover- we select randomly a crossover point and swap the tails of the two parents to form new off-springs.

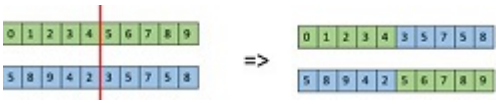


Figure 3.4: One point crossover: a random point in red followed by recombination

- Multi point crossover-is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.

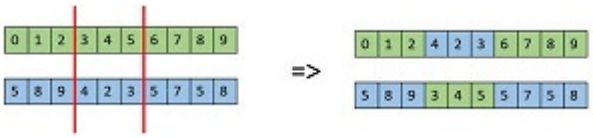


Figure 3.5: Multi point Crossover

- uniform crossover- the chromosome in this method are not divided up into segments, rather we treat each gene as a separate entity, essentially a probability is used to decide whether or not a chromosome will be included in the off-spring.

- Whole Arithmetic Recombination- commonly used for integer representation, takes the weighted average of the two parents by using the following formulae

$$Child1 = \alpha.x + (1 - \alpha).y$$

$$Child2 = \alpha.x + (1 - \alpha).y$$

if $\alpha = 0.5$, both children will be identical.

3.3.3 Survivor Selection-Elitism

This GA policy determines which individuals are to be kicked out and which are to be kept in the next generation. It ensures the fittest individuals are not kicked out of the population that is is they do not undergo mutation and are propagated as they are, and at the same time diversity should be maintained in the population.

GA's employ **Elitism** as the survivor selection policy, whose main principle is that *current fittest member of the population is always propagated to the next generation.*

Elitism in our ANN is defined as a percentage or a number.

I shall use the R package **GA** to implement the differential evolution algorithm in this research.

3.4 Performance Evaluation

Algorithm evaluation will be in terms of accuracy assessment that will be quantified by calculating the metrics below;

- Root Mean Squared Error (RMSE)- is an absolute measure of fit and is defined as the square root of the variance of the residuals, when applied it will indicate the absolute fit of the model to the data; how close are the observed data points to to the model predicted values, it is always in the same units as the response variable. The lower the values of RMSE the better the model fits data.

$$RMSE = \sqrt{\frac{1}{n}\sum_{x=1}^{n}\left(\frac{A_t - P_t}{A_t}\right)}$$

- Mean Absolute Error (MAE)- gives the average magnitude of the errors in a set of predictions, does this without considering their directions. That is, average over the test set of the absolute differences between predicted and actual observations with all individual differences having equal weight.

$$MAE = \frac{1}{n}\sum_{x=1}^{n}\frac{|A_t - P_t|}{|A_t|}$$

- Mean Squared Error (MSE)- the MSE of these estimator models will measure the average squared difference between predicted values and the actual values.

$$MSE = \frac{1}{n}\sum_{x=1}^{n}(A_t - P_t)^2$$

Where n is the number of predicted points, A_t is the actual recorded stock price and P_t is the predicted stock price value.

Prediction performance was evaluated through these performance measures; MAE, MSE, and RMSE with the interpretation being that the smaller the values of these measures

the closer the predicted time series values are to the actual values, and thus the better the model.

3.5 Data Description

The Data Set in use in this research is from the Nairobi Stock Exchange, NSE20 Share Index to be precise, the data is observed for a period of 10 years from January 2011 to January 2021, collected in 5 day weekly intervals with an exception of weekends and public holidays in the republic of Kenya.

From NSE20 share index data, I choose to work with the **East African Breweries** market, justification being that this company's derivatives have consistently been present on the stock exchange, and it's share prices over the 10 year period under review have been fairly consistent. The research data is comprised of 2610 observations of 5 variables, the response variable is the **eabshare_price**, while **eabmarket_value, eabshareprice_high, eabshareprice_low** are the explanatory variables.

3.5.1 Data Pre-Processing

3.5.2 Missing observations and imputations

The **mice** package from the R statistical software was used to data clean this data set by finding and imputing missing observations, approximately 226 observations were missing for 2 variables, 113 cases each.

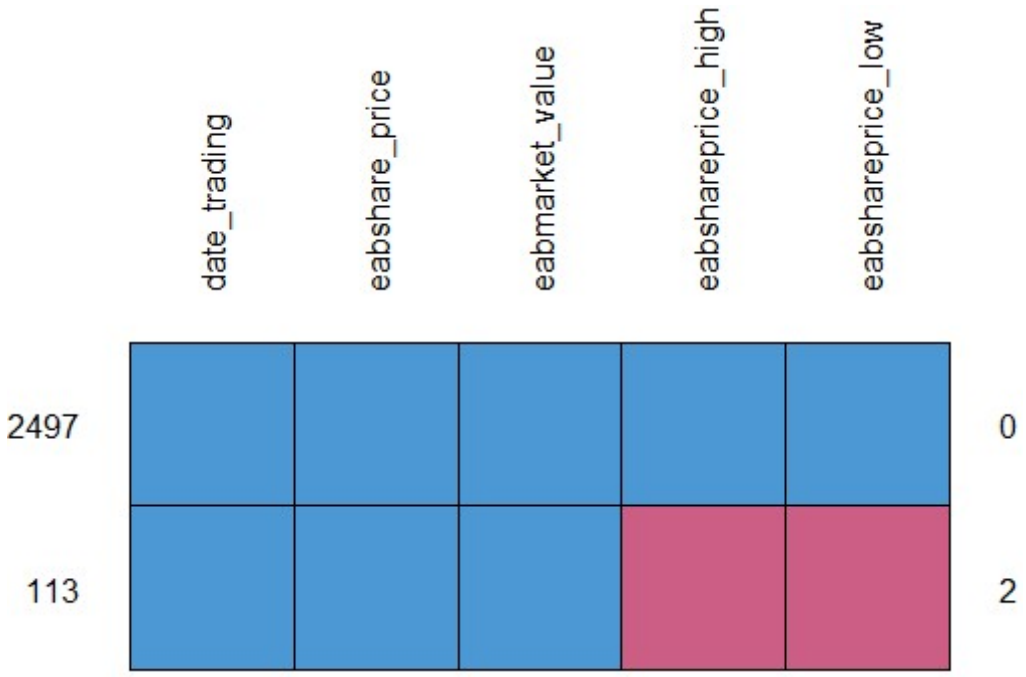


Figure 3.6: Graphical representation of missing observations

Using this package I imputed this values by applying the **predictive mean matching method**(PMM), which is a semi-parametric approach in that for each missing value it fills in a value randomly from among observed donor values from an observation who regression-predicted values are closest to the regression-predicted values for the missing value from the simulated regression model. PMM ensures imputed values are plausible. It does this by;

- forming a small set of candidate donors for each missing entry
- a donor value is then randomly selected/observed
- this "observed" value is then taken to replace the missing entry

This technique assumes the distribution of the missing values to be the same as that of the observed candidate donors with the main advantage of applying it being it's robustness against misspecification.

3.5.3 Data Normalization

A key step in ML algorithms involves data normalization, which bounds our ANN inputs to values between $[-1, 1]$ for this research the range is $[0, 1]$, subsequently making them comparable. Failure to normalize data will mean that the machine learning algorithm will be dominated by variables that use a larger scale which biases the model to inaccuracy, in addition, scaled data brings about the desired effect of faster convergence. Feature scaling normalization technique was applied to this data set through the R **Caret** package, as indicated in the formula below

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

It was chosen as the observed data is not Gaussian. On application of the technique, data was scaled to a fixed range of $[0, 1]$, and I ended up with data with a smaller standard deviation(S.D) which in-turn suppressed the effect of outliers.

3.5.4 Data Split

I used the *createDataPartition()* function from **Caret** package in R to split the data set into a training data set and a test data set at 70:30 percentage ration respectively.

Chapter 4

Results

The algorithms put forward were applied using R statistical software version 4.0.5 through it's Rstudio IDE version 1.4.1103, on a PC running Windows 10 Enterprise-64bit operating system(Build 19042), with a core-i5 processor and an 8GB RAM.

4.1 Results from Algorithms

4.1.1 Backpropagation model results

The neural network model was applied as follows, a configuration of one input layer was chosen with 3 nodes guided by the fact that we have 3 features, a hidden layer configuration of 3 nodes was chosen as it yielded a higher prediction accuracy as opposed to any other configuration, the *linear.output* argument was set to *True* with the assumption that their is a linear association between the independent and dependent variable, the *threshold* is set to 0.01% meaning if the change in error is less than 1% per iteration, then no further optimization was to be carried out.

I used 100 epochs and a Relu transfer function of the neurons of the hidden layer, whereas the linear transfer function was used for the single output layer since this is a regression problem.

The plot depicts result of the backpropagation of errors algorithm , a network with a single hidden layer and 3 neurons was the optimized network, it also highlight one of its drawbacks, in that, it took this model with a one network of a hidden layer of 3 neurons 1900 iteration steps to converge.

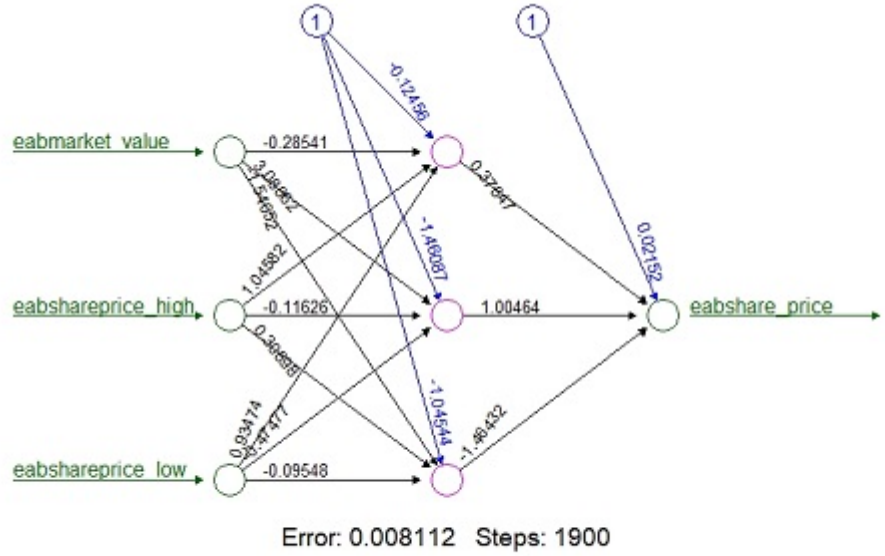


Figure 4.1: BPN Neural Network Configuration

Predicted prices versus actual observed prices for the market under study.

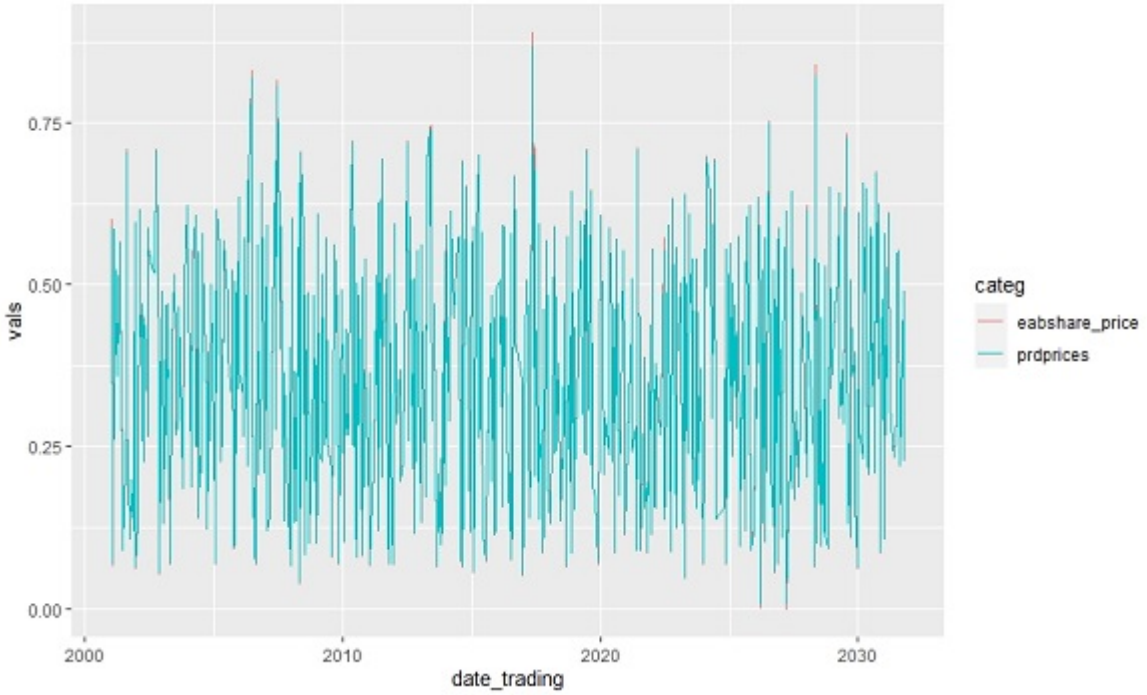


Figure 4.2: Real and Predicted EAB Share price values BPN Algorithm

The difference between actual and predicted share price values was too negligible, testament to this model's performance.

4.1.2 Genetic Algorithm

The genetic algorithm simultaneously performed feature selection and parameter optimization, for features, GA settled on all 3 features(number of features are negligible in this paper), 1830 samples were run from the training data set which took a considerable amount of time and gave me more or equal results compared to the BPN algorithm.
The main parameters were set as follows: maximum generations 30; population per generation 50; crossover probability is set to 0.8 with k-tournament selection method applied and chose the individual with the best fitness value after each iteration ; mutation probability at 0.1. GA was equipped with a random forest to investigate whether feature selection results in a marked performance difference, it did.
The best or optimal solutions are obtained at least at the 20^{th} generation coinciding with the 50^{th} iteration no improvement was noted after, so the algorithm stopped after the 30^{th} generation at 75^{th} iteration. See 4.3

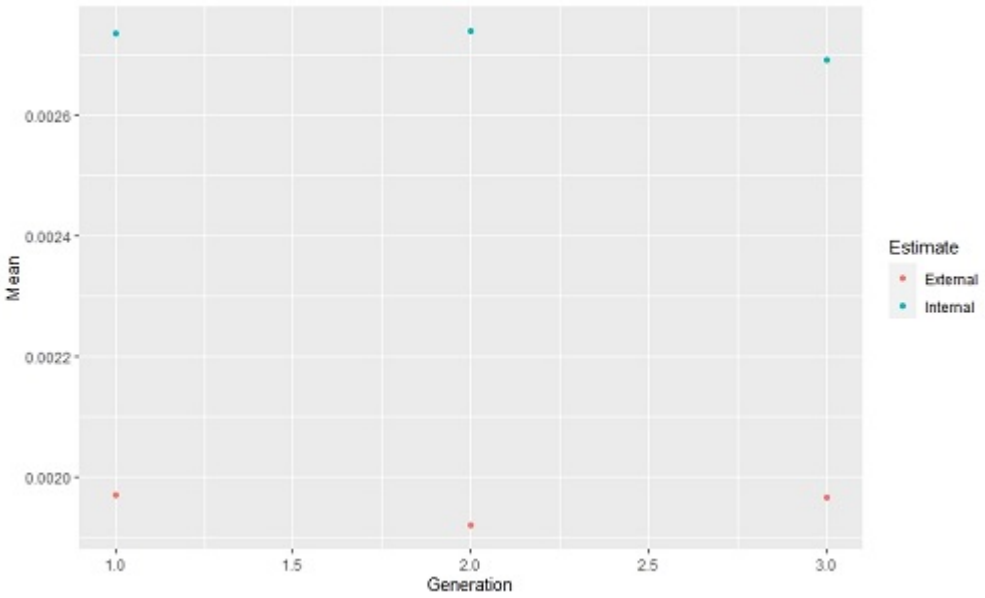


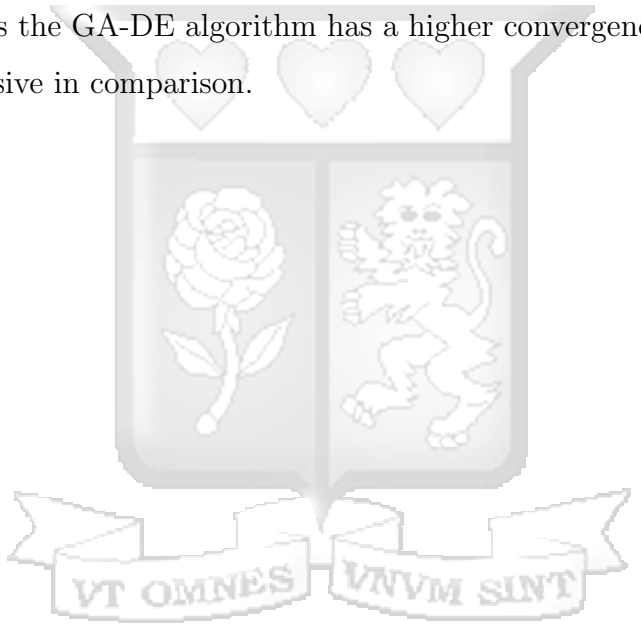
Figure 4.3: Algorithm Progression of the GA Object

4.2 Models compared

Prediction Model	MSE	RMSE	MAE	Accuracy %
BPN	0.0020	0.003	0.0016	99.1
GA-DE	0.0006	0.0025	0.0004	99.9

Table 4.1: Performance Metrics BPN GA-DE

The two algorithms were based on all indicators, figures in table 4.1 above, give an account of the differences between BPN and GA-DE time series forecasting models, the values are convergent. It is evident that machine learning algorithms are potent substitutes to traditional statistical methods, BPN method despite it's slight under performance versus the GA-DE algorithm has a higher convergence rate and is computationally inexpensive in comparison.



Chapter 5

Discussions, Conclusions and Recommendations

From literature a key variable in stock market prediction is the technical indicator, whose lack from the NSE20 share index may have greatly contributed to the small difference in performance between the algorithms with my hypothesis being that the GA-DE algorithm would have been superior for such a task.

Faced with a more complex data set, the advantages of the GA-DE would have been more apparent especially due it's ability to evolve a candidate solution by learning and subsequently picking out the most important features for regression.

Model accuracy and prediction power for BPN is affected by the selection in the number of layers in use in the hidden environment, a high numbered network has the ability of error reduction but with an increased convergence time, this in turn increases the complexity of the model and can result in a higher probability of model overfitting, special care should be taken while deciding on this, in addition the number of batches is crucial in regulating the speed and stability of the learning process which directly impacts accuracy, a high number of batches can lead to overfitting, rule of the thumb is to settle for model defined defaults.

In conclusion ANNs are excellent candidates for stock market prediction and their application by investors in the Nairobi Stocks Exchange is recommended and feasible with a high return on investment envisaged.

Bibliography

[1] Ahangar, R. G., Yahyazadehfar, M., and Pournaghshband, H. (2010). The comparison of methods artificial neural network with linear regression using specific variables for prediction stock price in tehran stock exchange. *arXiv preprint arXiv:1003.1457*.

[2] Anderson, D. and McNeill, G. (1992). Artificial neural networks technology a dacs state-ofthe-art report. *New York: Kaman Sciences*.

[3] Ariyo, A. A., Adewumi, A. O., and Ayo, C. K. (2014). Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112. IEEE.

[4] Ciaburro, G. and Venkateswaran, B. (2017). *Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles*. Packt Publishing Ltd.

[5] Fujita, H., Katafuchi, T., Uehara, T., and Nishimura, T. (1992). Application of artificial neural network to computer-aided diagnosis of coronary artery disease in myocardial spect bull's-eye images. *Journal of Nuclear Medicine*, 33(2):272–276.

[6] Gonzalez, R. T., Padilha, C. A., and Barone, D. A. C. (2015). Ensemble system based on genetic algorithm for stock market forecasting. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 3102–3108. IEEE.

[7] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[8] Gorman, R. P. and Sejnowski, T. J. (1988). Learned classification of sonar targets using a massively parallel network. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1135–1140.

[9] Holland, J. H. et al. (1992). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

[10] Hutchinson, J. M., Lo, A. W., and Poggio, T. (1994). A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889.

[11] Hwarng, H. B. (2001). Insights into neural-network forecasting of time series corresponding to arma (p, q) structures. *Omega*, 29(3):273–289.

[12] Inthachot, M., Boonjing, V., and Intakosum, S. (2016). Artificial neural network and genetic algorithm hybrid intelligence for predicting thai stock price index trend. *Computational intelligence and neuroscience*, 2016.

[Jena and Padhy] Jena, O. P. and Padhy, S. Application of ga with svm for stock price prediction in financial market.

[14] Lippmann, R. P. (1989). Review of neural networks for speech recognition. *Neural computation*, 1(1):1–38.

[15] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[16] Majhi, R., Panda, G., Sahoo, G., Dash, P. K., and Das, D. P. (2007). Stock market prediction of s&p 500 and djia using bacterial foraging optimization technique. In *2007 IEEE congress on evolutionary computation*, pages 2569–2575. IEEE.

[17] Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E., and Winkler, R. (1982). The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting*, 1(2):111–153.

[18] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[19] Mohapatra, P., Raj, A., and Patra, T. K. (2012). Indian stock market prediction using differential evolutionary neural network model. *International Journal of Electronics Communication and Computer Technology (IJECCT)*, 2(4):159–166.

[20] Nadia, M. (2019). Stock market price prediction system using neural networks and genetic algorithm. *Theoretical and Applied Information Technology*.

[21] Patel, J., Shah, S., Thakkar, P., and Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172.

[22] Pino, R., Parreno, J., Gomez, A., and Priore, P. (2008). Forecasting next-day price of electricity in the spanish energy market using artificial neural networks. *Engineering Applications of Artificial Intelligence*, 21(1):53–62.

[23] Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). Backpropagation: The basic theory. *Backpropagation: Theory, architectures and applications*, pages 1–34.

[24] Selvamuthu, D., Kumar, V., and Mishra, A. (2019). Indian stock market prediction using artificial neural networks on tick data. *Financial Innovation*, 5(1):1–12.

[25] Smith, L. N. (2017). Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE.

[26] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44.

[27] Warner, B. and Misra, M. (1996). Understanding neural networks as statistical tools. *The american statistician*, 50(4):284–293.

[28] Yizhen, L., Wenhua, Z., Ling, L., Jun, W., and Gang, L. (2011). The forecasting of shanghai index trend based on genetic algorithm and back propagation artificial neural network algorithm. In *2011 6th International Conference on Computer Science & Education (ICCSE)*, pages 420–424. IEEE.